PR #24197 完整报告

sgl-project/sglang

Refactor device timer, clean up metrics collector, and add fwd occupancy metric

合并时间：2026-05-02 01:25

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24197

执行摘要

- 一句话：重构设备计时器与指标收集器，新增前传占用率指标
- 推荐动作：值得精读，特别是设备计时器从调度器解耦到模型执行器的模式，以及指标收集器的按类重组方法。对于需要自定义指标的用户，需关注 emit_constants 的引入和旧环境变量的移除。

功能与动机

根据 PR 描述：metrics collector 原本有机增长，指标分散组织不佳，一次性常量与每步统计混在一起，计时器在重叠调度下不准确。本次重构旨在提高代码可维护性并引入更有效的 GPU 利用率信号。

实现拆解

1. 设备计时器重构：在 scheduler_metrics_mixin.py 中，将 record_bubble_metrics 等上下文管理器及相关调用移除；在 scheduler.py 的 init_model_worker 中，将 forward_pass_device_timer 注入到 model_runner 和所有 draft worker 的 device_timer 属性中。
2. 计时注入核心前向路径：在 model_runner.py 的 forward_decode 和 forward_extend 中，使用 device_timer.wrap(metadata={...}) 包装 self.model.forward() 调用；在 cuda_graph_runner.py 和 eagle_draft_extend_cuda_graph_runner.py 的 replay 方法中同样包装 self.graphs[key].replay()，使计时精确覆盖 GPU 执行部分。
3. 指标收集器清理：在 metrics_collector.py 中重新组织 SchedulerStats 字段，按“基础”、“内存池使用率”、“绝对 token 计数”、“推测解码”、“重召回”、“PD 分离”、“利用率”等分组，并添加详细内联文档；将 max_total_num_tokens、page_size、context_len 等一次性常量移到新的 emit_constants() 方法，替换旧的 emit_cache_config_info；移除 gpu_overlap_wait_seconds 相关方法。
4. 废弃代码移除：删除 environ.py 中的 SGLANG_DISABLE_REQUEST_LOGGING 和 USE_VLLM_CUTLASS_W8A8_FP8_KERNEL 环境变量，清理相关使用；从 tokenizer_manager.py 和 request_logger.py 中移除对应分支。
5. 新指标与测试：新增窗口式“fwd occupancy”指标（GPU 时间 / 墙钟时间），在 decode/prefill 日志中输出（需设置 SGLANG_ENABLE_METRICS_DEVICE_TIMER=1）；CI 测试 test_bench_one_batch_1gpu 验证单批 workload 的 p90 occupancy > 97.5%。

关键文件：

- python/sglang/srt/observability/scheduler_metrics_mixin.py（模块 指标层；类别 source；类型 core-logic；符号 init, _wrap_execution_reporter, record_forward_metrics, update_device_timer）：核心变更文件，重构设备计时器初始化与生命周期管理，移除 bubble 计时器，将 KvMetrics 改为 dataclass，缓存 graph_backend label，简化逻辑。
- python/sglang/srt/observability/metrics_collector.py（模块 指标层；类别 source；类型 core-logic；符号 get_histogram_conf_from_env, increment_gpu_overlap_wait_seconds, increment_forward_execution_seconds, increment_gpu_execution_seconds）：指标收集器全面整理：字段重新分组并添加详细文档注释，一次性常量发射移到 emit_constants，移除废弃代码（get_histogram_conf_from_env、num_running_reqs_offline_batch 等）。
- python/sglang/srt/model_executor/model_runner.py（模块 模型执行；类别 source；类型 data-contract；符号 model_is_mrope, _maybe_rebalance_after_rank_fault）：在 forward_decode 和 forward_extend 中注入 device_timer.wrap 以精确计时 GPU 前向调用；同时缓存 model_is_mrope 属性，提取 elastic EP 重组方法。
- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic）：删除旧 bubble 计时器调用，注入 device_timer 到模型运行器，使用 emit_constants 替换旧的 emit_cache_config_info。
- python/sglang/srt/model_executor/cuda_graph_runner.py（模块 模型执行；类别 source；类型 data-contract）：在 replay 方法中包装 device_timer 以捕获 CUDA 图执行时间，确保计时覆盖 GPU 回放。
- test/registered/perf/test_bench_one_batch_1gpu.py（模块 性能测试；类别 test；类型 test-coverage）：新增 fwd occupancy 验证测试，确保单批工况下 p90 占用率大于 97.5%，需要环境变量 SGLANG_ENABLE_METRICS_DEVICE_TIMER=1。

关键符号：record_forward_metrics, update_device_timer, record_bubble_metrics, cancel_bubble_timer, reset_device_timer_window, emit_constants, increment_forward_execution_seconds, forward_decode, forward_extend, replay, _maybe_rebalance_after_rank_fault

关键源码片段

python/sglang/srt/observability/scheduler_metrics_mixin.py

核心变更文件，重构设备计时器初始化与生命周期管理，移除 bubble 计时器，将 KvMetrics 改为 dataclass，缓存 graph_backend label，简化逻辑。

```
@dataclasses.dataclass
class KvMetrics:
    """KV 缓存指标，使用 dataclass 提供默认值和类型提示。"""
    request_active_slots: int = 0
    request_total_slots: int = 0
    kv_active_blocks: int = 0
    kv_total_blocks: int = 0
    num_requests_waiting: int = 0
```

```
    gpu_cache_usage_perc: float = 0.0
    gpu_prefix_cache_hit_rate: float = 0.0
    data_parallel_rank: int = 0


class SchedulerMetricsMixin:
    def init_metrics(self: Scheduler, tp_rank: int, pp_rank: int, dp_rank: Optional[int]):
        # ... 其他初始化 ...
        self.stats = SchedulerStats()
        # 缓存图形后端标签，避免每步重复构建字符串
        self._graph_backend_label = {
            "cpu": "cpu graph",
            "npu": "npu graph",
            "musa": "musa graph",
        }.get(getattr(self, "device", ""), "cuda graph")
        self.enable_metrics = self.server_args.enable_metrics
        self.is_stats_logging_rank = self.attn_tp_rank == 0
        self.current_scheduler_metrics_enabled = self.enable_metrics and (
            self.is_stats_logging_rank or self.server_args.enable_metrics_for_all_schedulers
        )
        self.enable_mfu_metrics = False
```

python/sglang/srt/observability/metrics_collector.py

指标收集器全面整理：字段重新分组并添加详细文档注释，一次性常量发射移到 emit_constants，移除废弃代码（get_histogram_conf_from_env、num_running_reqs_offline_batch 等）。

```
@dataclass
class SchedulerStats:
    # Basics
    num_running_reqs: QueueCount = field(default_factory=QueueCount)
    num_queue_reqs: QueueCount = field(default_factory=QueueCount)
    num_grammar_queue_reqs: int = 0
    gen_throughput: float = 0.0
    cache_hit_rate: float = 0.0
    decode_sum_seq_lens: int = 0

    # Memory pool usage ratios (0.0–1.0).
    # Each pool tracks: used = total - available - evictable, usage = used / total.
    #
    # token_usage: max(full, swa, mamba) — the bottleneck across all pools.
    # FIXME: misleadingly named "token_usage"; rename requires API deprecation.
    # full_token_usage: full-attention KV cache pool usage (always active).
    # swa_token_usage: sliding-window attention KV cache pool usage (hybrid SWA models only,
    e.g. Gemma2).
    # mamba_usage: Mamba SSM state pool usage (hybrid SSM models only, e.g. Jamba).
    token_usage: float = 0.0
    full_token_usage: float = 0.0
    swa_token_usage: float = 0.0
```

```
    mamba_usage: float = 0.0

    # Absolute token counts for the full-attention KV cache pool.
    # Invariant: kv_available_tokens + kv_evictable_tokens + kv_used_tokens <= max_total_num_
    tokens
    # (the gap accounts for protected/session-held tokens not exposed here).
    # max_total_num_tokens is emitted once at startup via emit_constants.
    #
    # kv_available_tokens: free (unallocated) slots in the pool.
    # kv_evictable_tokens: slots holding radix-cached KV data that can be evicted for new
    requests.
    # kv_used_tokens: actively used slots (locked by running requests). Equals full_num_used.
    # num_used_tokens: max(full_num_used, swa_num_used) for hybrid-SWA models, else full_
    num_used.
    # Does NOT include the mamba pool.
    num_used_tokens: int = 0
    kv_available_tokens: int = 0
    kv_evictable_tokens: int = 0
    kv_used_tokens: int = 0
```

python/sglang/srt/model_executor/model_runner.py

在 forward_decode 和 forward_extend 中注入 device_timer.wrap 以精确计时 GPU 前向调用；同时缓存 model_is_mrope 属性，提取 elastic EP 重组方法。

```
  def forward_decode(
      self,
      forward_batch: ForwardBatch,
      skip_attn_backend_init: bool = False,
      pp_proxy_tensors=None,
  ) -> Union[LogitsProcessorOutput, PPProxyTensors]:
      # 设置额外参数
      if not skip_attn_backend_init:
          if hasattr(self.model, "prepare_forward_batch"):
              self.model.prepare_forward_batch(forward_batch)
      # ... 其他准备 ...
      kwargs = {}
      if self.support_pp:
          kwargs["pp_proxy_tensors"] = pp_proxy_tensors

      # 如果启用了设备计时器，则包装 GPU 前向调用以精确计时
      ctx = (
          self.device_timer.wrap(metadata={"category": "decode"})
          if self.device_timer
          else contextlib.nullcontext()
      )
      with ctx:
          return self.model.forward(
              forward_batch.input_ids,
              forward_batch.positions,
```

```
            forward_batch,
            **kwargs,
        )
```

评论区精华

本 PR 未产生 review 评论。提交历史中可见若干迭代：将 GPU 繁忙百分比钳制在 100%、重命名 available_gpu_memory_gb 为 startup_available_gpu_memory_gb、将 gpu_execution_seconds_total 重命名为 forward_execution_seconds_total，以及多次修复指标准确性。这些迭代反映了对指标命名一致性和边界条件的关注。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 回归风险：删除 bubble timer 和 gpu_overlap_wait_seconds 可能影响依赖这些指标的监控，但新指标 fwd occupancy 更准确且默认不启用。
  - 性能风险：device_timer.wrap() 作为上下文管理器仅在启用时生效，对未启用环境变量的用户无额外开销。
  - 兼容性风险：移除 SGLANG_DISABLE_REQUEST_LOGGING 等环境变量可能需用户更新配置；PR 未提供迁移路径。
  - 测试覆盖：CI 仅覆盖单批场景，多批复杂调度下的 occupancy 准确性未充分验证。
- 影响：
  - 用户影响：需要设置 SGLANG_ENABLE_METRICS_DEVICE_TIMER=1 以启用新占用率指标；移除的环境变量可能影响旧配置。
  - 系统影响：指标收集更结构化，新增的 fwd occupancy 为 GPU 利用率提供新视角，有助于诊断调度效率。
  - 团队影响：代码更易维护，但需要适应新的指标分类和命名。
  - 风险标记：核心路径变更，废弃环境变量移除，新指标默认禁用

关联脉络

- 暂无明显关联 PR